BiDAF with Explicit Token Linguistic Features

Stanford CS224N Default Project

Mentor: Lauren Zhu

Sifan Ye
Department of Computer Science
Stanford University
sifanye@stanford.edu

Abstract

This project aims to explore the effectiveness of extending input token features in BiDAF [1] model to improve on SQuAD v2 QA task [2]. The extension is inspired by existing approaches in open domain QA and neural translation and is motivated by reading comprehension techniques I have learned as a human who doesn't speak English as first language. The extra features include part-of-speech (POS) tag, named entities (ENT) tag, sentiment and lemma match. The addition of those extra features has shown significant performance improvement over the baseline BiDAF model [1], scoring EM/F1 score of 61.9/65.1 on the validation set and 59.4/62.1 on the test set.

1 Introduction

Question Answering (QA) is a challenging task in natural language processing (NLP), as it tests the reading comprehension ability of the machine system. This includes understanding the information provided in the context, and extracting such information when presented a question. With the introduction of the SQuAD v2 QA dataset [2], a further challenge is presented to the machine system to determine that the context does not have the necessary information to answer the presented question. This requires the machine to build a stronger correspondence between the question and the context.

In this project, I explored the effectiveness of providing explicit linguistic features from the spaCy [3] English model, with the overall goal to improve performance on the SQuAD v2 QA task [2] over the baseline BiDAF model [1] without character-level embeddings. I experimented with various methods to encode such labels as input and different methods to join those with the word embeddings input. I found that providing the labels as one-hot vector and concatenating those to the word embedding provides the most performance boost. I also found that although the model is able to have improved performance, it also suffers from some overfitting on the explicitly provided linguistic features.

2 Related Work

In open domain QA tasks, one approach is to read Wikipedia as source for factoid questions. [4] In this approach, proposed by Danqi Chen et al., tokens in a paragraph are encoded with a feature vector which includes the word embedding of the token, an exact match feature which is a boolean feature denoting if the token can be matched to some form of a word in the question, and token features including part-of-speech (POS), named-entity-recognition, and normalized term frequency. The experiments involving performance with varying inclusion of those features show that those additional explicit features are very helpful at improving performance.

To improve performance of language models trained on large corpra on knowledge based task, Zhengyan Zhang et al. proposes to use a linear combination of multi-headed attention over word encodings and entity encodings to incorporate information from knowledge graphs. [5] The method have also shown significant performance improvement when extra linguistic features are provided on top of word embeddings.

3 Approach

3.1 Baseline

The baseline model for this project is a BiDAF [1] implementation with only word-level embeddings, using pretrained GloVe word embeddings [6]. The baseline model is trained for 30 epochs, with hidden size 100, and evaluates on the validation set at:

F1: 60.44, EM: 57.03, NLL: 03.12

3.2 Motivation

As a human learning English as a second language, similar challenges often come up in tests such as SAT and GRE, where we would need to discern, in the given options for answers to a question, which has complete and necessary information from the given passage. Since many of those tasks are presented under time constraint, one effective method to tackle those problems is to skim through the text and when reading the question, find correspondences in the text, and quickly rule out options that don't match with the text. Another challenge would be reading comprehension with unseen words. One method to tackle those is to infer its role in the sentence by its surrounding words' POS, and infer its sentiment by its surrounding words' sentiment. With a rough interpolation of the potential meaning of the word, correspondence can be made with the question for an answer.

One of the main challenges in the SQuAD v2 QA task [2] is discerning if sufficient information is provided in the context to answer the question. This usually takes the form of the question having few different key words than the context and unmentioned words in the context. Since using explicit linguistic features have shown performance improvement in other NLP tasks [4] [5], and is used in human reading comprehension, this project aims to improve performance of the baseline BiDAF model by providing similar linguistic features.

3.3 Tokenizer Linguistic Features

In the baseline model, the spaCy [3] English model is used for tokenization during data preprocessing. Using the same model during data processing, I obtained several linguistic features of each token. `pos_`, `ent_type_` are strings representing POS and named entity type (ENT) labels on the token. Those are then mapped to an index set where no label is indexed at 0 for tokens in the context and in the question. `sentiment` is a float representing the sentiment of the token, provided for tokens in the context and in the question. `lemma` is an integer label representing the lemma form of the token. By comparing `lemma` of the token in the context with the `lemma` of each token in the question, I obtain a boolean feature for the context tokens only, represented by a 0 or 1 integer, called *lemma match* denoting if the lemma of a context token can be matched to the lemma form of some token in the question. Those are then stored alongside the word embeddings for the dataset to be used as inputs to the model.

3.4 Incorporating Linguistic Features

3.4.1 Encoding POS and ENT

Since POS and ENT are integer labels, they need to be provided to the model as some form of embedding. In this project I compared training embeddings of various dimensions from scratch during training for the QA task and frozen one-hot vector encoding of those indices while the 0 index representing an empty label is represented as a zero-vector. Experiments show that one-hot vector encoding provides better performance increase. Details in section 4.

3.4.2 Incorporating into word embeddings

To incorporate the linguistic features to the word embeddings to be encoded by the highway encoder layer [7] to be fed to the RNN layer, I compared between concatenation and linear combination.

In concatenation (see Figure 1), the vectors for POS and ENT and the values of sentiment and lemma match are directly concatenated to the word embedding vector. The resulting vector of is then projected down to the word embedding size using a linear layer with no bias to be fed to the highway encoder.

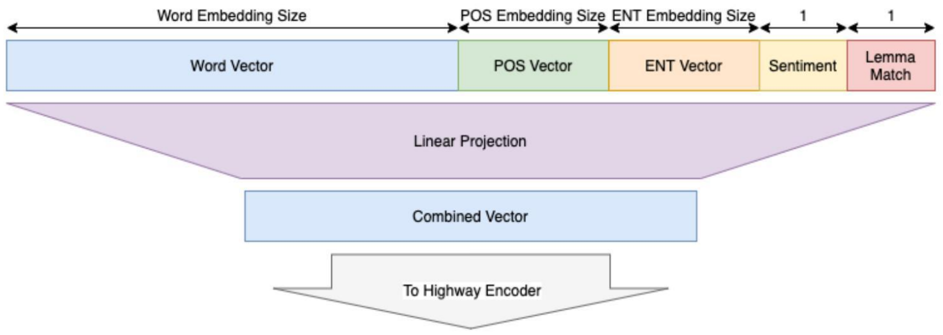


Figure 1: Concatenating Features to Context Word Vector

In linear combination (see Figure 2), each feature is projected to a vector of the same length of the word embedding using a linear layer without bias, then summed to the word vector to be fed to the highway encoder.

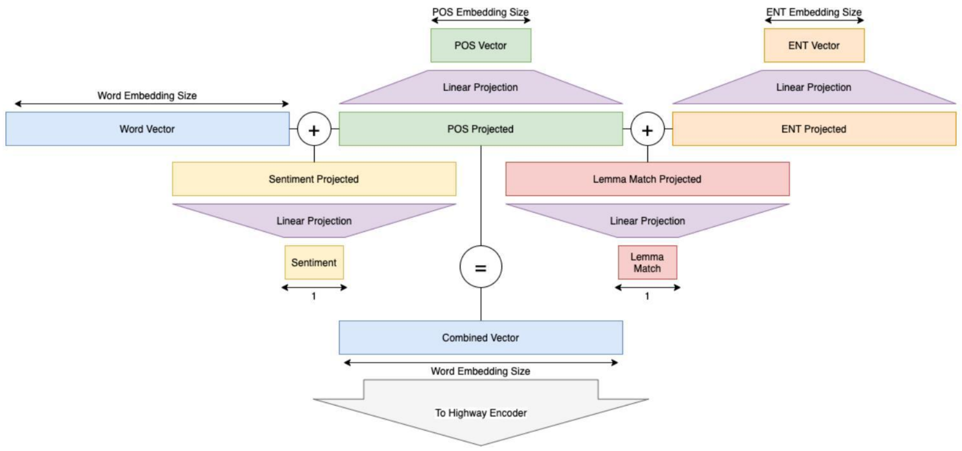


Figure 2: Linear Combination of Context Word Vector and Features

Experiments show that generally concatenation works better, with the most effective combination shown to be concatenation with one-hot vectors. Details in section 4.

3.5 Bottleneck

Another commonly used technique to combat overfitting is a bottleneck layer, where a feature vector is projected down to a smaller size then projected up to the desired hidden size for the model to learn a compressed feature. In this project, a bottleneck layer (Figure 3) is applied after combining the vectors before the highway encoder and yielded some performance improvement. Details in section 4.

3.6 Dropouts on Linguistic Features

Dropout is a common regularization technique used in neural networks to combat overfitting on input features and in the baseline BiDAF, dropout is applied to pretrained word vectors. By comparing the performance of applying dropouts to different additional features, we observe that the model is overfitting on the extra provided linguistic features. Consistent results on validation and test set is achieved when dropouts are applied to all features. Details in section 4.

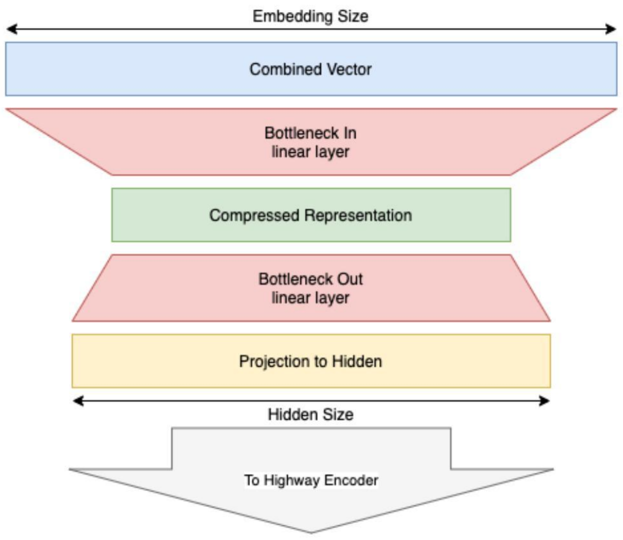


Figure 3: Bottleneck layer before highway encoder

3.7 Hidden Size

I have chosen to use hidden size 80 instead of the default size of 100 as another method to combat overfitting by reducing the number of parameters in the model. I have not tried and compared lower hidden sizes since 80 performs well and that there is a time constraint on the project.

4 Experiments

4.1 Data

The dataset used for training is the training split of the provided SQuAD v2 QA dataset [2] and the dev split is used for evaluation. The pretrained word embeddings used for the model are the GloVe vectors [6]. Linguistic features for the tokens are provided by the spaCy [3] English model.

4.2 Evaluation method

The baseline and the models are evaluated on the validation split of the provided dataset on 3 metrics: EM, F1 and NLL.

4.3 Experimental details

All experiments ran for 30 epochs which the same learning rate at 0.5 and the same dropout rate at 0.2.

4.3.1 Encoding POS and ENT, and combination method

Since POS and ENT has similar total number of labels (21 and 20 respectively), I experimented with the same learned embedding size for both. Table 1 shows the performance of different embedding sizes when the features are combined using linear combination. Table 2 shows the performance of different embedding sizes when the features are combined using concatenation. Both experiments are conducted at hidden size 80 without bottleneck.

From those two sets of experiments, we can see that generally the longer the embedding length the better the performance. This can be attributed to the fact that a higher dimensionality is able to encode more information about the labels and their relations to each other. However, we can also observe that frozen one-hot encoding performs the best, potentially due to the fact that no extra training from scratch is involved and the information and relation with other features can be encoded in the weights used to combine with other vectors.

Embedding Length	F1	EM	NLL
baseline	60.44	57.03	3.12
8	61.73	58.4	3.12
10	61.8	58.6	2.9
12	62.25	59.18	2.86
16	62	58.88	2.97
one-hot	62.46	59.6	2.74

Table 1: Performance of Different Embedding Sizes when Linearly Combined

Embedding Length	F1	EM	NLL
baseline	60.44	57.03	3.12
12	64.73	61.49	2.72
16	64.9	61.82	2.73
one-hot	65.84	62.71	2.63

Table 2: Performance of Different Embedding Sizes when Concatenated

Those two sets of experiements, we can also observe that concatenation performs better than linear combination. One possible explanation for this is that concatenation uses less number of weights and hence is less prone to overfitting and vanishing gradients.

4.3.2 Bottleneck Size

To control for overfitting, I experiemented with a bottleneck layer between the combined vector and the highway encoder, at hidden size 80, feature embedding size 16 and using linear combination. (Table 3)

Bottleneck Size	F1	EM	NLL
baseline	60.44	57.03	3.12
No Bottleneck	62	58.88	2.97
60	63.28	60.33	2.69
50	62.42	60.21	2.73

Table 3: Performance of different bottleneck sizes

The limited experiment shows that adding a bottleneck of size 60 gives the best performance given the condition. Assuming that this generalizes to concatenation, I have fixed the bottleneck size at 60 for later experiments.

4.3.3 Dropouts on Linguistic Features

To combat overfitting I have experimented dropouts on different combination of features. The experiments are run with hidden size 80, concatenation of linguistic features and bottleneck of size 60. (Table 4)

Although dropout on POS and ENT only gives the best result on the validation test, the same combination produced a much lower score on the test set, suggesting that there is heavy overfitting on some of the features. For comparison, I submitted the test set results for the other two combinations, with scores in Table 5.

Comparing performance on both the validation and the test set, we can see that the model is likely heavily overfitting on the explicit sentiment and the lemma match feature. With dropouts on all features, I see that the performance on the validation and test set is more consistent. Since I did not compare the results of no dropout on POS and ENT, I cannot draw a conclusion on the extent of overfitting on those 2 features. However, this does show the importance and effectivness of using dropout as a regularization technique.

Dropout	F1	EM	NLL
baseline	60.44	57.03	3.12
POS and ENT	65.2	62.26	2.66
POS, ENT and Sentiment	65.09	61.91	2.72
All	62.78	59.72	2.75

Table 4: Performance of Different Dropout Combinations, on validation set

Dropout	F1	EM
POS and ENT	45.374	40.490
POS, ENT and Sentiment	62.114	59.442
All	61.064	57.853

Table 5: Performance of Different Dropout Combinations, on test set

4.4 Results

From Table 4 and 5 we can see that providing the model with extra linguistic features is effective in improving performance of the model on the validation set over the baseline. However, the test results on the test set suggests that the model also overfits on those explicit features and dropouts need to be applied to all features to control overfitting. Since I do not have the performance of the baseline model on the test set, I cannot draw any conclusions about the effectiveness of the provided linguistic features on the test set. From the quantitative results, I can only conclude that it is likely that the addition of pretrained explicit linguistic features provides a inexpensive way to improve the performance of the model but requires regularization to be effective on generalizing the use of such features.

5 Analysis

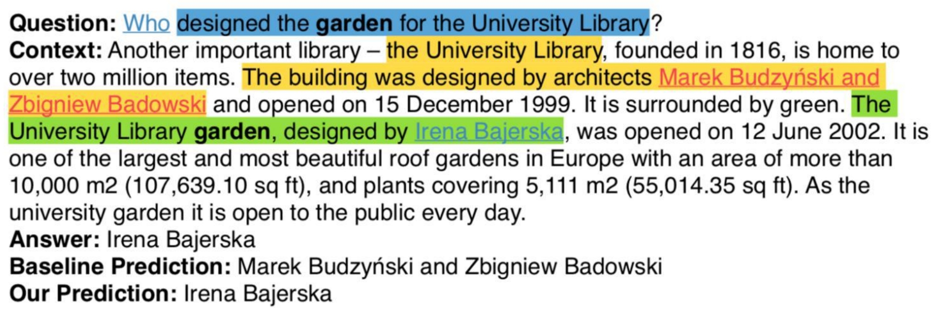
From the validation set I identified 3 questions, each of a different type, where my model answered correctly that the baseline model did not. In each figure:

- Question keywords are highlighted in blue
- Close context match is highlighted in yellow
- Exact context match is highlighted in green
- "wh-" question word is underlined and colored blue
- Correct match to the "wh-" word is underlined and colored blue
- Wrong match to the "wh-" word is underlined and colored red
- Important keywords are **bolded**

Those examples demonstrate that with inclusion of explicit linguistic features, my model is better able to correctly find question context correspondence, or the lack thereof, to return a correct answer for the question, compared to the baseline model, on the validation set.

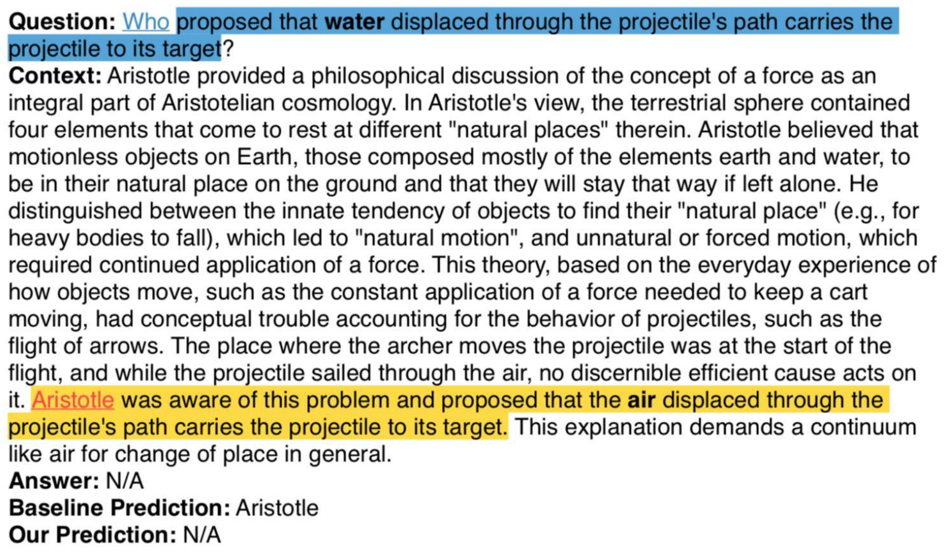
5.1 Missed Keyword

Figure 4 shows a wrong answer potentially caused by a missed keyword in the question. The baseline correctly identified the designers of the *University Library* and matched "who" to two names. However, the question is actually asking about the **garden**, where "**garden** for the *University Library*" in the quesiton is analogous to "*University Library* **Garden**". Here, my model successfully identified the designer of the building actually asked for in the question. In reading comprehension, it is important to match every word from the question to the context to obtain an accurate answer and this shows that the included linguistic features, potentially lemma match, allowed my model to correctly match all keywords in this sample question.



Question: Who designed the **garden** for the University Library?
Context: Another important library – the University Library, founded in 1816, is home to over two million items. The building was designed by architects Marek Budzyński and Zbigniew Badowski and opened on 15 December 1999. It is surrounded by green. The University Library **garden**, designed by Irena Bajerska, was opened on 12 June 2002. It is one of the largest and most beautiful roof gardens in Europe with an area of more than 10,000 m2 (107,639.10 sq ft), and plants covering 5,111 m2 (55,014.35 sq ft). As the university garden it is open to the public every day.
Answer: Irena Bajerska
Baseline Prediction: Marek Budzyński and Zbigniew Badowski
Our Prediction: Irena Bajerska

Figure 4: Sample Question 1: Missed Keyword



Question: Who proposed that **water** displaced through the projectile's path carries the projectile to its target?
Context: Aristotle provided a philosophical discussion of the concept of a force as an integral part of Aristotelian cosmology. In Aristotle's view, the terrestrial sphere contained four elements that come to rest at different "natural places" therein. Aristotle believed that motionless objects on Earth, those composed mostly of the elements earth and water, to be in their natural place on the ground and that they will stay that way if left alone. He distinguished between the innate tendency of objects to find their "natural place" (e.g., for heavy bodies to fall), which led to "natural motion", and unnatural or forced motion, which required continued application of a force. This theory, based on the everyday experience of how objects move, such as the constant application of a force needed to keep a cart moving, had conceptual trouble accounting for the behavior of projectiles, such as the flight of arrows. The place where the archer moves the projectile was at the start of the flight, and while the projectile sailed through the air, no discernible efficient cause acts on it. Aristotle was aware of this problem and proposed that the **air** displaced through the projectile's path carries the projectile to its target. This explanation demands a continuum like air for change of place in general.
Answer: N/A
Baseline Prediction: Aristotle
Our Prediction: N/A

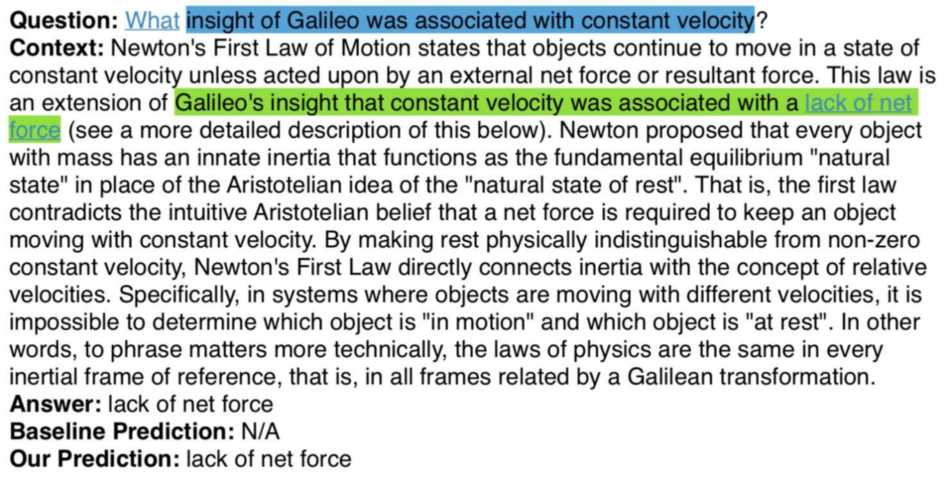
Figure 5: Sample Question 2: Miss Matched Keyword

5.2 Mismatched Keyword

Figure 5 shows a wrong answer potentially caused by a miss matched keyword in the question. This is a typical adversarial example of an unanswerable question, since one changed key word of the question means that the sentence cannot be matched exactly to the context and when no other relevant information is provided, no answer from the context can be provided. Here the baseline model successfully identified the very similar sentence in the context, and correctly identified that Aristotle corresponds to who in the question. However it did not recognize that **air** keyword from the context is replaced by **water** in the question, making the entire sentence a spurious match. My model correctly recognized that **air** does not match to **water** in the question and hence cannot provide an answer from the context given.

5.3 Missed Relation

Figure 6 shows that the baseline model wasn't able to provide an answer when an answer is available in the context. One possible reason is that the baseline model is looking for "associated with constant velocity" in the question but wasn't able to recognize that "associate" is a symmetric relation and "*a* associates with *b*" is equivalent to "*b* associates with *a*". This may have caused the baseline model to unable to make a question context correspondence to extract the correct answer. My model was able to recognize constant velocity can be matched to the context before "associated with" and the noun phrase after it is the correct answer matching to "what".



Question: What insight of Galileo was associated with constant velocity?
Context: Newton's First Law of Motion states that objects continue to move in a state of constant velocity unless acted upon by an external net force or resultant force. This law is an extension of Galileo's insight that constant velocity was associated with a lack of net force (see a more detailed description of this below). Newton proposed that every object with mass has an innate inertia that functions as the fundamental equilibrium "natural state" in place of the Aristotelian idea of the "natural state of rest". That is, the first law contradicts the intuitive Aristotelian belief that a net force is required to keep an object moving with constant velocity. By making rest physically indistinguishable from non-zero constant velocity, Newton's First Law directly connects inertia with the concept of relative velocities. Specifically, in systems where objects are moving with different velocities, it is impossible to determine which object is "in motion" and which object is "at rest". In other words, to phrase matters more technically, the laws of physics are the same in every inertial frame of reference, that is, in all frames related by a Galilean transformation.
Answer: lack of net force
Baseline Prediction: N/A
Our Prediction: lack of net force

Figure 6: Sample Question 3: Missed Relation

6 Conclusion

In this project, I have successfully provided explicit token linguistic features to a modified BiDAF model to achieve better performance on SQuAD v2 QA task [2], scoring EM/F1 score of 61.9/65.1 on the validation set and 59.4/62.1 on the test set.

Motivated by question context correspondence in my experience with reading comprehension, and inspired by similar works in NLP, supplying token linguistic features has shown to be able to increase performance, especially when strict correspondence is needed to provide a correct answer. My approach involves encoding indexed features as one-hot vectors and concatenating those to the token word vector. The combined vector is then passed through a bottleneck before passing to the highway encoder to control for overfitting. The use of those features doesn't require extensive extra time on training and only increases data preprocessing time which is a one-time cost. So when compared to the baseline model, where those linguistic features can be hidden in the pretrained word vectors and the question context relation can be learned as parameters of the model, adding explicit features provides an efficient way of improving performance. However, as they are not learned from scratch, those features are also very prone to overfitting, which needs to be carefully controlled in the model. As I have learned in a hard way, regularization cannot be overlooked when adding or learning extra features.

Due to time constraint on the project, I did not have the oppurtunity to conduct complete controlled experiments for comparison of different combination methods, different encoding methods and different regularization methods. Hence further work may include designing complete experiments for a more rigorous comparison between the methods tested in this project. Another potential experiment is to look into the true effectiveness of the additional features, by designing experiments with randomly generated features for each token. By comparing the performance with mostly correct labels and randomly generated placebo labels, we can check if it was the features that improved the performance, or just the fact of extra weights that improved the performance. Other further work may also include finding a most efficient set of linguistic features to include while minimizing overfitting effect since POS, ENT and sentiment are not the only features of a token.

As with the problem with overfitting on the current model, further work may include modifying the attention layer to learn to better extract question context correspondence instead of using an explicit match feature. Another limitation is that the current model still only considers the start and end positions of the answer in context independently, where they should considered jointly when making correspondence with the question. Hence further work may include incorporating conditioning between the start and end position prediction, in junction with the context question correspondence.

References

[1] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension, 2018.

[2] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

[3] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020.

[4] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *CoRR*, abs/1704.00051, 2017.

[5] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy, July 2019. Association for Computational Linguistics.

[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[7] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.